# Samovar : An evaluation framework for real time applications deployment over WSANs

Lionel Havet
INRIA Nancy-Grand Est
54603 Villers-les-Nancy
lionel.havet@loria.fr

Adrien Guenard
INRIA Nancy-Grand Est
54603 Villers-les-Nancy
adrien.guenard@loria.fr

Françoise Simonot-Lion
University of Nancy,
LORIA (UMR 7503)
54506 Vandoeuvre-les-Nancy
francoise.simonot@loria.fr

## Abstract

*Wireless Sensor and Actuator Networks (WSANs) combine sensors and actuators interconnected by wireless networks in order to perform distributed sensing and acting tasks. Closed-loop controllers can therefore be deployed on WSANs; such systems have to meet specific requirements in terms of performance, dependability, energy and cost which raises great challenges due to the unreliability of wireless communications. A way to ensure that a system meets the required properties is to model it and go through its analysis. Building a model requires both deep knowledge on the system as well as on the used framework. Therefore there is a need for frameworks well-suited to the targeted systems and to the properties to verify. We propose an approach meeting these conditions and a simulation framework, Samovar, based on Matlab / Simulink, allowing the modeling of the network protocols (Mac and routing services) and the resources sharing policy thanks to the TrueTime toolbox. Several classes of components (application, nodes, networks and middleware) and a clear semantics for their composition are identified. Furthermore, the design of Samovar was also driven by the need to transfer easily software components model between the concrete systems and its simulated model. The modeling and simulation method as well as the Samovar framework are illustrated on a Pursuit Evasion Game.*

## 1 Introduction

Intelligent building, home automation, transportation, disaster relief operations and more generally cyber-physical systems (CPS) are all critical applications having stringent safety requirements, relying on various closed-loop controls. Such applications are foreseen to be deployed on Wireless Sensor and Actuator Networks (WSANs). WSANs applications show many advantages: distributed control, equipment of hazardous or difficult to access environments, fully mobile operation, low costs. But WSANs give real challenges for control application development due to the unreliability of the wireless network. The Quality of Service (QoS) delivered by the network is not guaranteed and may not match the QoS expected by the application. Applications therefore typically suffer from variable packet delivery delays and packet loss. Applications can already cope with such situations using fallback functional mode, adapting control laws, etc. On the other side a network can adapt itself to the requirements of application by making some oversampling for instance. A middleware between the application and the network service is the right place to deal with QoS in WSANs applications balancing simultaneously between application and network adaptability to meet applications requirements. As mentionned in [1] QoS challenges in middleware are still an open issue for research.

Modeling applications deployed over WSANs is necessary for the *a priori* validation, deployment optimization and fine tuning and sizing in terms of safety, resources usage and performance. WSAN based applications deal with continuous time processes, time based controls and event based network protocols. Their *a priori* analysis requires to model them thanks to state-transitions machines able to represent their continuous aspects: hybrid automata, hybrid Petri Nets, etc. But the mathematical and exhaustive exploration of such hybrid model could be very costly and even impossible for large scale applications. Therefore a simulation approach dealing with continuous and discrete aspects appears currently as the best pratice for WSAN applications *a priori* evaluation. It is a cost effective solution, the development is made easier and scenarios can be repeated. However, whatever are the analysis technique, mathematical or simulation one, an important issue in the modeling activity is the accuracy of the model, meaning in the context of the paper, that if the properties are proved on the model, then they will be met by the concrete system. A second issue is to minimize the distance between the model and the concrete implementation of the system; a way that contributes to achieve this goal is to use in the model

the same software components than the ones that will be deployed in the concrete system. Moreover, such principle facilitates the development process by an easy transfert from the modeled system down to the concrete one.

To the best of our knowledge, most of the simulators developped for WSAN applications evaluation deal either with continuous models of the system and its control functions or with discrete models of wireless network protocols. Therefore, there is definitely a need for a co-simulation framework. Network performance indeed affects the stability and safety of the controlled physical systems while the dynamics of the physical system and the limited embedded resources influence the deployment and the configuration of the WSAN.
In this paper, we propose a framework, Samovar [2], for the modeling and evaluation of the operational architecture of adaptive systems deployed on WSANs.

- The framework allows the evaluation and sizing of WSANs applications through simulation by providing relevant indicators :
  - Application indicators : Quality of Control, safety, etc.
  - Underlying execution architecture indicators : energy, end-to-end delays, packet losses, deadline overrun, etc
- This framework takes into account:
  - The software application components.
  - The performance and services provided by the hardware architecture (scheduling, protocols, execution/transmission time, etc).
  - The middleware responsible for application and network adaptation.
  - The controlled and monitored system environment (dynamics, behavior, etc).
  - The hazards impacting networks or electronic components.
- The framework offers high fidelity to the real system especially by providing code reuse between the modeled system and the real one.
- Based on our experience [3] and in order to make the transition to real physical devices easier, the elements modeled in the Samovar framework are not decoupled from real devices.

In its current version the framework contains the models of Khepera III robots [4] and MICA Z motes [5]. These simple devices already allow the design of various applications dealing with robotics and WSN [6, 7, 8]. The figure 7 shows, for example, a typical Pursuit Evasion Game (PEG) system that will be the context of the testbed presented in the section 5.

The section 2 makes a survey of existing simulation tools bringing partial solution to the targeted problem. In the section 3 we detail the design choices made for Samovar. The elements composing the Samovar framework are presented in the Section 4. Finally the section 5 demonstrates the possibilities of the Samovar framework on a case study.

## 2 State of the art

As stated in the former section, the proposed simulation framework targets systems that, on the one hand, integrates components relying on continuous models (for example, dynamics and control of autonomous mobile robots, dynamics of the environment) and, on the other hand, are relevant of discrete event systems (such as scheduling policies, network protocols, resource sharing policies, etc.) Therefore a co-simulation approach is definitely needed to take into account both aspects.

Currently, several tools address only one of the aforementioned points: dynamics of the controlled systems (autonomous mobile robots, in the current version of Samovar or more generally physical systems), network and protocol; the cooperation between discrete event simulators and continuous physical systems simulators has also already been proposed in the literature. In this section a quick review of existing simulation and modeling frameworks is proposed.

### 2.1 Robotic simulation

In the targeted applications, actors are, for example, intelligent robotic agents interacting with the physical system. Moreover the robotics field is impacted by the emerging technology of WSN that provides it with an augmented environment. On the other side, robotics is already a science of cyber-physical integration. Unlike the network simulation frameworks, there are no emerging simulation framework [9, 10]. Although these frameworks support multi-robot interactions, they do it only at a functional level. In particular, simulation runs under the assumption of infinite resources (processors and networks); if it is sometimes possible to introduce a delay between a perception and an action in order to model the execution time of a task running the decision algorithm and the message passing between agents, none of them provide realistic communication scheme amongst robots neither a realistic model of a multitasking system. Therefore integrating these points leads to the conclusion that only co-simulation with one of these robotics oriented frameworks has to be considered. On this point of view the publicly available Player/Stage project [11] and the Cyberbotics Ltd. company [12] propose frameworks that both provide an interface to envisage co-simulation.

### 2.2 Network simulation

Comparisons amongst a large number of publicly available or commercial simulation tools are the topic of

several recent reviews of WSN simulation frameworks [13, 14]. These surveys share few simulation frameworks which are emerging amongst the others, but they do not share common criteria for the framework's evaluation.
There are three de facto frameworks for network simulation: the network simulators NS-2, OMNET++, and OPNET suite of products.
The Network Simulator, NS-2, is a high quality network simulation framework in which many newly proposed MAC protocols are evaluated [15]. Therefore it supports many of the networking standards (TCP / UDP, routing and multi-cast protocols). In the context of WSAN, the native NS-2 tool suffers first from low possibilities for mobility modeling. Mobility of nodes must be predefined in a scenario and can actually not be driven by the system behavior as this simulator does not allow the modeling of continuous behavior. Second, this framework does not provide any means for the simulation of the continuous evolution of an environment as it is measured by a wireless sensor application. Typical class of modeled applications are network applications such as CBR, FTP, HTTP and Telnet. Its successor NS-3 has an improved design but no improvements are foreseen so far regarding modeling of mobility or continuous behaviors.
OMNET++ framework shows low mobility possibility as well, similarly defined in scenarios. Modeling of sensor data for stimulating the WSN does not exist.
OPNET is rather similar to NS-2 and OMNET++ in terms of functionalities; it is less targeted for early stage evaluation of new protocols and it is much more suited for network application deployment. Dynamic models and control algorithms of systems can be designed in OPNET but this is rather tedious with OPNET Proto-C language .
The lack of suited functionalities for WSNs of these three general network simulation frameworks led to new simulators geared towards the WSNs. A first one is TOSSIM [16] which compiles directly from TinyOS code. But TOSSIM does not provide any mean to introduce mobility of sensor or sensor stimulation within the simulation. A further extension, WorldSense [17], proposes a limited mobility model. A third one Prowler [18] is designed within the Matlab [19] environment. The simulator was originally written to simulate Berkeley MICA motes. Unfortunately its design does not allow benefiting from the co-simulation framework of Matlab/Simulink.

### 2.3 Physical system simulation

WSN are converging towards cyber-physical systems seamlessly integrating communication, sensing, control and physical systems. Physical system simulation frameworks must simulate modern products that integrate, e.g., electric, mechanic, hydraulic components as well as their controller whose behavior is described by continuous functions.
In this scope there are two leading simulation frameworks: Matlab/Simulink and Modelica. Both frameworks provide a wide library to model various kind of physical systems. For our concern in WSAN modeling, none of these tools natively supports the modeling and the simulation of network protocols or local schedulers.
The Matlab/Simulink framework supports the definition of functions decribing the evolution of a component in C/C++ language; this is an interesting aspect in the scope of source code reuse on real targets. In this scope Matlab features the Real Time Workshop Embedded Coder which generates automatically C code from the Simulink blocks and can also achieve target code generation (Matlab Target Compiler).

### 2.4 Hybrid simulation

**Co-simulation.** In order to overcome the issues described so far in network simulation frameworks, many proposals arise in combining one of these network simulation frameworks with a physical modeling framework. Matlab/Simulink and Modelica are two very powerful tools for modeling systems and implementing control algorithms. However, they have limitations in simulating computer networks. Here are some of the co-simulation frameworks: OPNET and Matlab [20], NS-2 and Modelica [21], PICCSIM [22] relying on NS-2 and Simulink. These co-simulation frameworks apparently take advantage of each individual framework, but some remarks can be addressed regarding time synchronization. First, simulation is always driven by one of the two frameworks which can lead to missed particular simulation steps on the ”slave” framework. Second, nodes only have one supporting execution architecture which can not be split over two frameworks: communication functions, sensing, actuating are all running on one processor.
To synchronize two simulation tools during a simulation, one dealing with the modeling of the continuous world and the other one targeting the modeling of the discrete event systems (protocols, scheduling policies, etc.) leads us to conclude that a better solution is to extend one of this tool in order to model all the features. The literature proposes such approach consisting in extending a discrete event based framework.
**Discrete framework extension.** The Agent/Plant extension of NS-2 [23] allows the simulation of physical systems dynamics and controllers along the network, but still it is not possible to handle the mobility of a node through such a dynamic system.
The Castalia framework [24] developed on top of OMNET++ shows low mobility possibility as well, mobility is always predefined in scenarios. Castalia addresses the problem of simulating sensor data for stimulating the WSN but in a rather limited fashion. These solutions are still lacking capability in terms of continuus simulation. The last alternative is the extension of a continuous system simulator.
**Continuous framework extension.** The support of WSN modelling in continuous framework can typically be achieved with the specific toolbox TrueTime [25] available in both Matlab/Simulink and more recently Model-

ica. TrueTime was developed for network and embedded systems modeling and simulation. This toolbox offers the TrueTime kernel block which allows the simulation of the resource sharing policies on a computer node and several blocks modeling network protocols as the TrueTime wireless network block which simulates the physical and medium access control layer (MAC) of wireless network. TrueTime does not support as many protocols or MAC or higher layer protocols (at network, transport OSI layer, for example) as NS-2, but it is really seen as a promising tool for co-simulation of WSAN [26].
As a conclusion, Matlab/Simulink, thanks to the TrueTime toolbox and the Matlab Target Compiler is a good candidate to achieve all the requirements defined in the introduction, this will be the basis of the proposed Samovar framework.

## 3 Samovar design principles

As stated in the former section, a simulation framework based on Matlab/Simulink and on the TrueTime toolbox can achieve the requirements we imposed:

- the modeling of continuous functions (physical environment, controllers) and of resource sharing policies (scheduling on a node and protocols).
- the modeling of software components in C/C++ language; this allows to reuse this code for realizing the practical application.
- the integration of significant probes allowing to monitor indicators that are relevant for the verification of properties required by the targeted applications.

Therefore the framework Samovar is built upon TrueTime toolbox in the Simulink framework. Let us focus on the second requirement given in the above list; it leads to define a common metamodel as well as the suited methodology that is usable for the simulation model and for the software architecture deployed on the practical platform. For the applications that we intend to deal with, two kinds of nodes are identified:

- the sensor nodes (MICA Z nodes) which will support application software components, middleware local components and, for some of them, routing algorithms; as they communicate thanks to a wireless network, their behaviour is also constraint by the medium access policy (MAC protocol) and the physical layer;
- the mobile nodes (Khepera robots) that support application software components, middleware local components, routing algorithms; as for the sensor nodes, the message sending and / or reception of such a node obeys to the underlying protocol of the wireless network (MAC protocol and physical layer); moreover, the mobility of these nodes as well as the sensing of their environment is ensured thanks to a set of low layer functions (sensor reading functions, low level controllers).

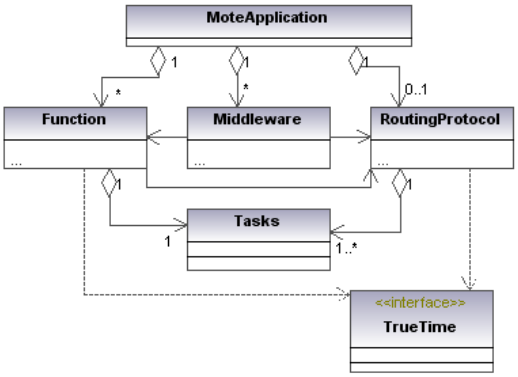


**Figure 1. UML class diagram for Mote software. A mote software is made of a routing protocol and applications.**

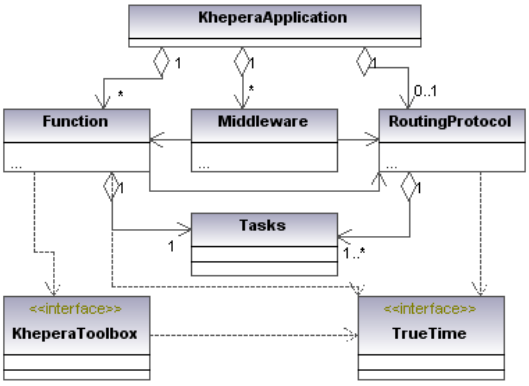


**Figure 2. UML class diagram for Khepera III software. The software is made of routing protocol and applications possibly using the Khepera III toolbox.**

The figure 1 and the figure 2 illustrate the metamodels proposed for the design of the simulation model as well as for the realization of the practical application. Encapsulated in the TrueTime interface are the scheduling local policy and the wireless low layer protocols (MAC sublayer and physical layer); they are instanciated, on the one hand, in the simulation model by Simulink TrueTime blocks and, on the other hand, in the practical platform by the low layer of the communication stack implemented on each node. A similar principle is used for the modeling of the sensing and mobility control of the Kheperas. The sensing and mobility control functions on the Khepera robots is achieved through the Khepera III toolbox [27] for which an abstraction is provided in the simulation model. As a

result robot control functions use the same library interface in the simulation model and on the practical application.
The software components whose source code can be ”transfered” between the simulation model and the real platform are only the application components, the local middleware components and the software components realizing the routing protocol algorithm. All these components are developped in C/C++ language.
The exchanges of messages between distant nodes or the broadcast of messages from a node are done through the routing protocol component. A mailbox models the strategy of the transfert between routing protocol components and application components. This model respects the strategy that is implemented on the practical platform.
Finally, the active objects, the tasks responsible of the software component execution on a processor and the messages exchanged through a network, have to be identically deployed in the simulation model (TrueTime inputs) and on the practical platform (Tiny OS tasks on the MICA Z nodes or Linux threads on the Kheperas robots).

# 4 Model elements

## 4.1 Wireless network

The wireless network model used in the Samovar Framework is based on a TrueTime block. This model is depicted on figure 3. It handles the Medium Access Control (MAC) and the Physical layer (PHY). The network block handles all the frame sending requests by queuing them and blocking them in the queue for a simulated delay which includes backoff time CTS/RTS if any, frame data and acknowledgement if any. After this delay, if the frame transmission is judged successfull (no interference, good signal to noise ratio, etc) the frame is delivered to the Matlab / Simulink model of the target nodes. The original TrueTime network model has been extended to support probabilistics pertubations and to provide some new indicators : message delivery ratio (ratio of the successfully transmitted packets to the total sent packets) and message delivery average delay.

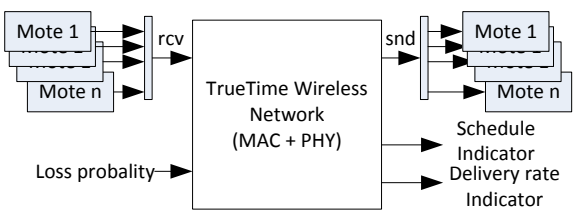


**Figure 3. Samovar wireless network model.**

## 4.2 Samovar mote model

The mote model encapsulates a TrueTime kernel block as shown on figure 4. The mote model executes a set of tasks following a configurable scheduling policy. The set of tasks which can be executed must be defined in the TrueTime kernel *init* script. In the Samovar Framework this *init* script is generated automatically by defining the functions, the middleware and the routing protocol object instances of the mote software metamodel.
The environment interface provides analog sensor data to mote tasks. The model allows to disturb the scheduler execution with variable execution time or clock drift. The mote model provides some indicators : energy consumed, tasks schedule. Similarly to the network model, a custom delivery ratio has been added.
Modeling a WSN involving hundreds of nodes can not be achieved by simply dragging hundreds of blocks in a Simulink model. To overcome this issue the Samovar platform provides a Mesh component that can be instanciated and configured in order to automatically generate the required set of nodes according to predefined topologies.

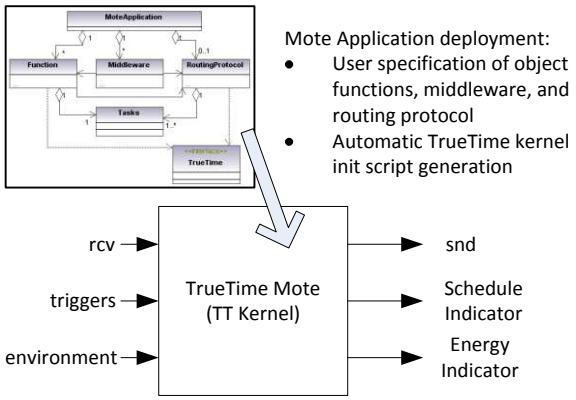


**Figure 4. Samovar mote model.**

## 4.3 Khepera III robot

The figure 5 details the model of the Khepera III robot and its controller.

**Khepera model** The Khepera model includes a first block matching the robot dynamics: The robot is a differential wheeled mobile. A unicycle model is used to represent its kinematic characteristics. Speed and acceleration saturation algorithms are used to represent mechanical constraints (maximum speed of 0.35 $m/s$ and maximum acceleration of 0.6 $m/s^2$) and to adapt commands to have the desired behavior. The second block is the sensor block; it models the acquisition of measures provided by the modeled environment. This model is enriched by non functional parameters as noises and acquisition delay. Modeling the behaviour of the I2C bus, embedded in the robot, at a fine grain, would lead to handle a large number of events during the simulation and therefore would be time consuming. Therefore, we chose to abstract it only by delays.
**Khepera controller** The Khepera controller model is a TrueTime Kernel block executing a set of tasks according to a given scheduling policy. These tasks basically

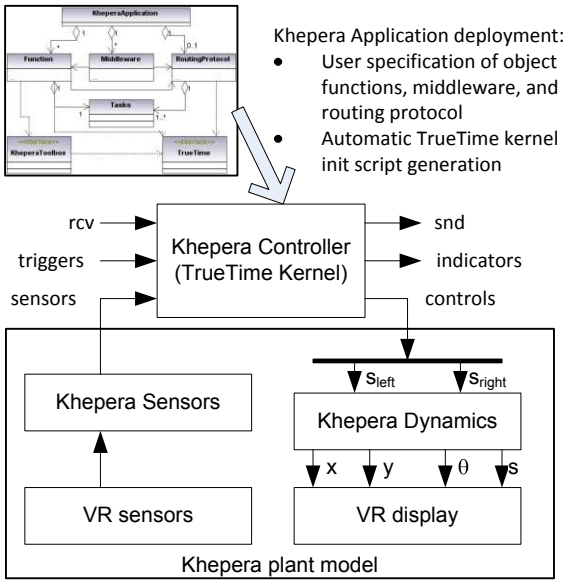


**Figure 5. Robot plant and robot controller model. The robot plant model is decoupled from the controller so that it can be reused.**

use sensor information or network message to control the robot. As for the mote, the TrueTime Kernel block tasks set is automatically defined by instanciation thanks to the metamodel.

### 4.4 Simulink library

The figure 6 presents the Simulink building blocks included in the Samovar library.

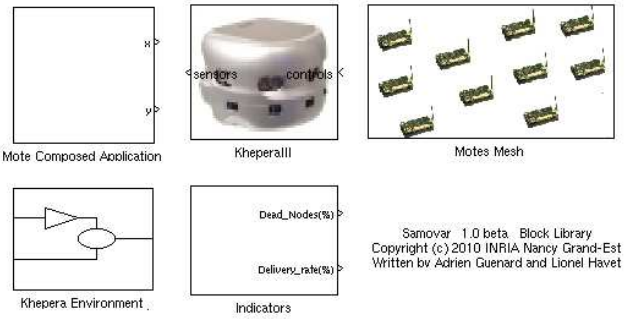


**Figure 6. Samovar Simulink library. An example of model built by assembling such elementary blocks is illustrated in figure 8**

## 5 Sample Application

To demonstrate the capabilities of the Samovar framework, an example application is presented in this section. It is a multi-robot pursuit evasion game (PEG) [28, 6].

### 5.1 Test-bed description

For this benchmark PEG application, the scenario is the following: a target robot moves within a uniform Wireless Sensor Network. A second robot called pursuer tries to catch the first one and once it gets close enough to the target robot, the target robot is caught. Both robots periodically ($T_{track}$) broadcast a tracking message over the wireless network. Each mote receiving this message sends the robot's message Received Signal Strength Indicator (RSSI) to a base station using a multi-hop protocol. The base station calculates the robots positions with all the received RSSI messages: it first converts RSSI to distances using an estimated radio propagation model and then, using a triangulation lateration algorithm, it computes each robot position. At last, the base station sends the target position and the pursuer position to the pursuer robot over a second wireless network. From this information the pursuer robot computes a command aimed at tracking the target robot. Simultaneously to localization information transmission, each mote of the wireless network takes part of an area monitoring: each mote periodically ($T_{area}$) sends sensor data to the base station (for example pressure sensor in the floor, temperature, pressure, humidity...). The figure 7 illustrates this application. The application performance criteria is defined by the time needed by the pursuer to catch the target robot.

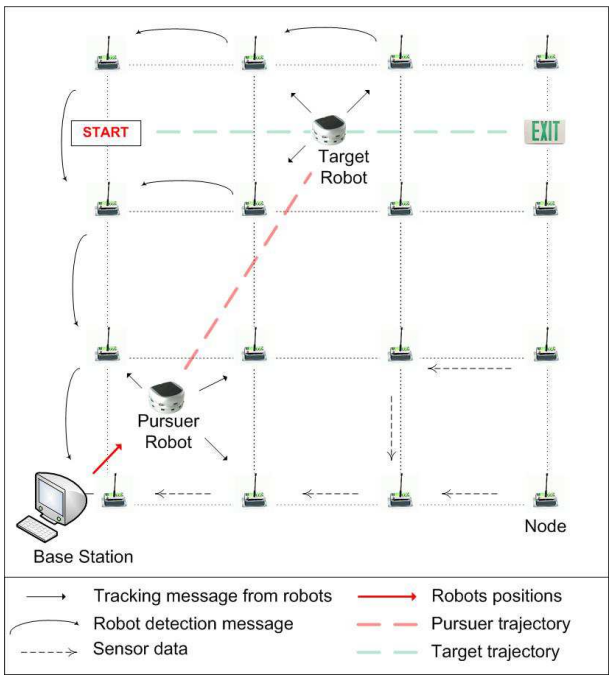


**Figure 7. PEG test bed description**

### 5.2 Experimentation and results

The PEG benchmark has been modeled in the Samovar framework. The Simulink model is shown on the figure 8. An experiment is setup to know under which area monitoring traffic load the application performance criterion is

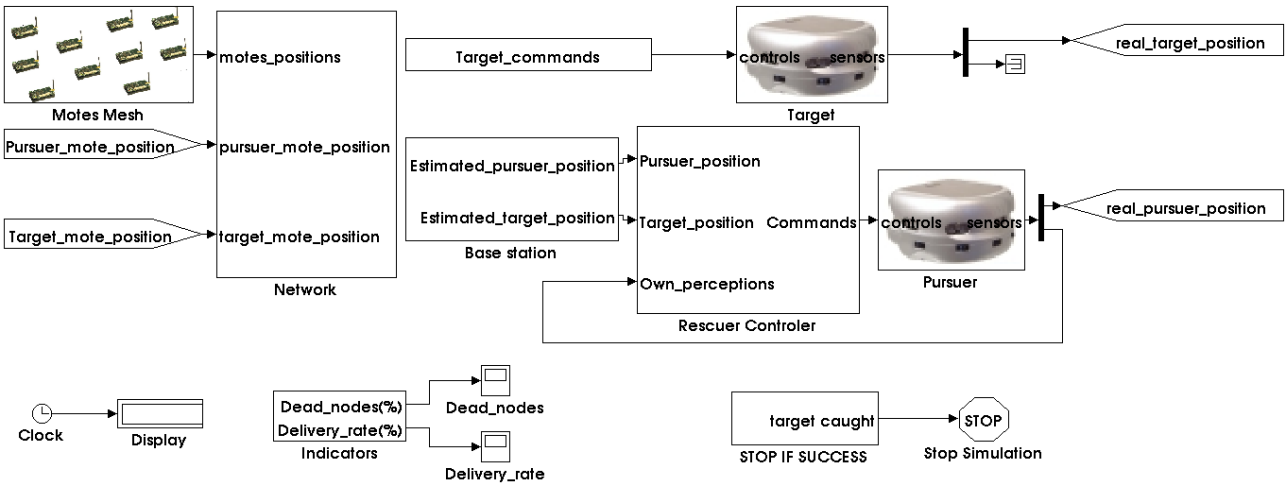


**Figure 8. Matlab/Simulink PEG model**

not met.

**Experiment setup** The target robot follows a predefined path scenario in order to constantly reproduce the same target robot moves. The wireless sensor network is composed of sixteen ZigBee/IEEE 802.15.4 motes uniformly distributed in a 150*150 $cm^2$ square area: small distances between motes allow fast mobility of the robots. They communicate via multi-hop paths using AODV (Ad hoc On Demand Distance Vector) routing protocol. The network dedicated to the communication between the base station and the pursuer robot uses Wifi/IEEE802.11b technology. Since there is only two motes in this Wifi network, delays are insignificant here and then the communication is just represented by a wire in the Matlab/Simulink model.

**Measurements and results** The area monitoring period, $T_{area}$, of each mote is changed in a range from $10ms$ to $100ms$ by steps of $10ms$, with a robot tracking period $T_{track}$ of $100ms$. For each $T_{area}$, the time needed by the pursuer to catch the target is reported on figure 9. As a comparison, the time to catch the target robot with infinite network resources (infinite bandwidth, no noise, no loss) is given. The experiments were all repeated ten times, in average an experiment lasts $2min$. for $10s$ simulation time on a Core2 CPU@1.86Ghz.

Since network load increases when $T_{area}$ decreases, it is no surprise to see that the pursuer does not manage to catch the target, see figure 9. The time required to catch the target using infinite resources on the network is also given as a comparison. The transitional region happens below a threshold $T_{crit}$ of $20ms$ for $T_{area}$. In fact, below this threshold too many tracking messages sent by the robots and forwarded by the WSN do not reach the base station. To monitor this loss we define a delivery ratio $d_r$ as the number of tracking message received by the base station divided by the number of tracking message emitted. The values of $d_r$ are given for different $T_{area}$ values on figure 10. The figure shows that for $T_{area}$ lower

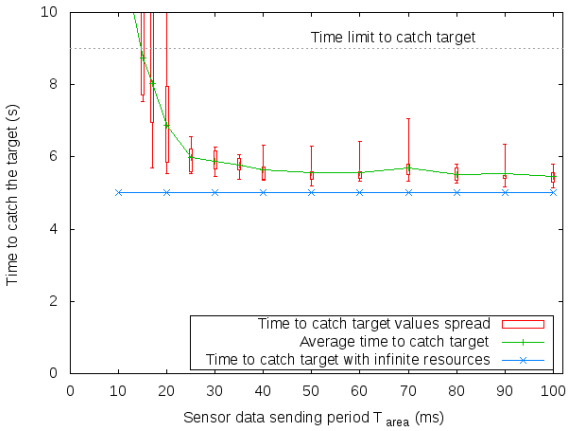


**Figure 9. Application performance (time to catch the target) for different area monitoring period $T_{area}$**

than $T_{crit}$ the delivery ratio $d_r$ goes below 3 which is the theoretical limit to perform a triangulation lateration algorithm.

In order to always satisfy the application criterion, a simple design choice for the application is to set $T_{area}$ above the $20ms$ threshold. An other design possibility is to use the delivery ratio $d_r$ as a QoS indicator. This is a useful hint for an adaptive middleware design for the application : a middleware could fine tune the $T_{area}$ of each node based on the delivery ratio, accepting message loss on the network when location of robot is required maintaining the delivery ratio $d_r$ above the critical threshold of 3. When no robot location is required the area monitoring period can be lowered.

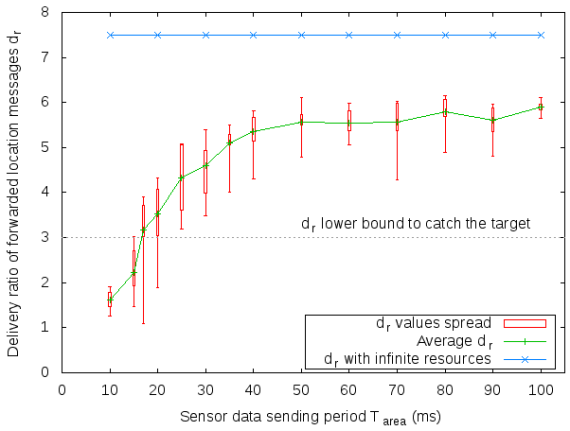


**Figure 10. Delivery ratio for different value of $T_{area}$**

## 6 Conclusions

This paper outlines important modeling aspects regarding the deployment of a critical application over a Wireless Sensor Network. The Samovar framework first release demonstrated the capabilities of the framework for simulation of typical WSAN applications. Samovar is available at *samovar.loria.fr*. Future work concerning the Samovar Framework will address first the implementation of other routing protocol for mobile devices and second the generation of an adaptive middleware for the PEG testbed presented in this paper.

## References


[1] Y. Li, C. S. Chen, Y.-Q. Song, and Z. Wang. Real-time QoS support in wireless sensor networks: a survey. In *7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems - FeT'2007*, Toulouse France, 2007.

[2] Samovar, http://samovar.loria.fr.

[3] L. Havet and F. Simonot-Lion. Timing properties requirements and robustness analysis of a platoon of vehicles. In *8th IFAC International Conference on Fieldbuses and networks in industrial and embedded systems - FeT2009*, Ansan Korea, May 2009.

[4] K-Team, http://www.k-team.com.

[5] Crossbow Technology, http://www.xbow.com.

[6] Z. Li, H. Li, F. Zhang, J. Chen, and Y. Sun. An indoor sensor network system for pursuit-evasion games. In *4th International Conference on Mobile Ad-hoc and Sensor Networks*, dec. 2008.

[7] J. Takahashi, K. Sekiyama, and T. Fukuda. Cooperative object tracking with mobile robotic sensor network. In *9th International Symposium on Distributed Autonomous Robotic Systems*. Springer Berlin Heidelberg, 2009.

[8] K. Luthy, E. Grant, and T. Henderson. Leveraging rssi for robotic repair of disconnected wireless sensor networks. In *IEEE International Conference on Robotics and Automation*, April 2007.

[9] J. Craighead, R. Murphy, J. Burke, and B. Goldiez. A survey of commercial & open source unmanned vehicle simulators. In *IEEE International Conference on Robotics and Automation*, Roma, Italy, April 2007.

[10] J. Kramer and M. Scheutz. Development environments for autonomous mobile robots: A survey. *Autonomous Robots*, 22(2):101–132, 2007.

[11] http://playerstage.sourceforge.net.

[12] O. Michel. Cyberbotics ltd. webots tm : Professional mobile robot simulation. *Int. Journal of Advanced Robotic Systems*, 1:39–42, 2004.

[13] M. Korkalainen, M. Sallinen, N. Karkkainen, and P. Tukeva. Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications. *The Fifth International Conference on Networking and Services*, 2009.

[14] M. Jevtic, N. Zogovic, and G. Dimic. Evaluation of wireless sensor network simulators. In *17th Telecommunications forum TELFOR 2009*, November 2009.

[15] N. Boughanmi and Y.-Q. Song. A New Routing Metric for Satisfying Both Energy and Delay Constraints in Wireless Sensor Networks. *J. Signal Process. Syst.*, 51(2):137–143, 2008.

[16] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *1st international conference on Embedded networked sensor systems*, New York, NY, USA, 2003. ACM.

[17] A. Fraboulet, G. Chelius, and E. Fleury. Worldsens: development and prototyping tools for application specific wireless sensors networks. In *6th international conference on Information processing in sensor networks*, Cambridge, Massachusetts, USA, 2007.

[18] http://www.isis.vanderbilt.edu/projects/nest/prowler.

[19] Mathworks, http://www.mathworks.com.

[20] H. M. Shahidul, Y. Hongnian, A. L. Griffiths, and T. C. Yang. Co-simulation framework for networked control systems over multi-hop mobile ad-hoc networks. In *17th IFAC World Congress, 2008, Vol. 17, Part 1*, Seoul, South Korea, July 2008.

[21] A. Al-Hammouri, V. Liberatore, H. Al-Omari, Z. Al Qudah, M. S. Branicky, and D. Agrawal. A co-simulation platform for actuator networks. In *5th international conference on Embedded networked sensor systems*. ACM, 2007.

[22] http://autsys.tkk.fi/en/Control/PiccSIM.

[23] M. S. Branicky, V. Liberatore, and S. M. Phillips. Networked control system co-simulation for co-design. In *American Control Conference*, Denver, CO,USA, 2003.

[24] http://castalia.npc.nicta.com.au/index.php.

[25] A. Cervin, M. Ohlin, and D. Henriksson. Simulation of networked control systems using TrueTime. In *3rd International Workshop on Networked Control Systems: Tolerant to Faults*, Nancy, France, June 2007.

[26] A. T. Al-Hammouri, M. S. Branicky, and V. Liberatore. Co-simulation Tools for Networked Control Systems. In *11th international workshop on Hybrid Systems*, St. Louis, MO, USA, 2008.

[27] http://khepera3toolbox.svn.sourceforge.net/.

[28] R. Severino and M. Alves. Engineering a search and rescue application with a wireless sensor network - based localization mechanism. Los Alamitos, CA, USA, 2007. IEEE Computer Society.